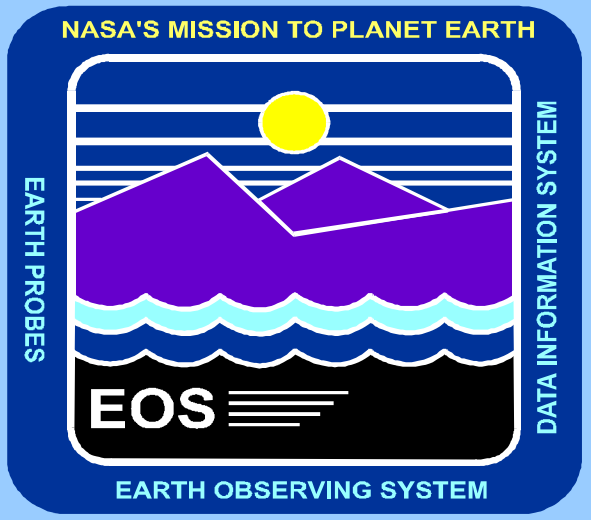
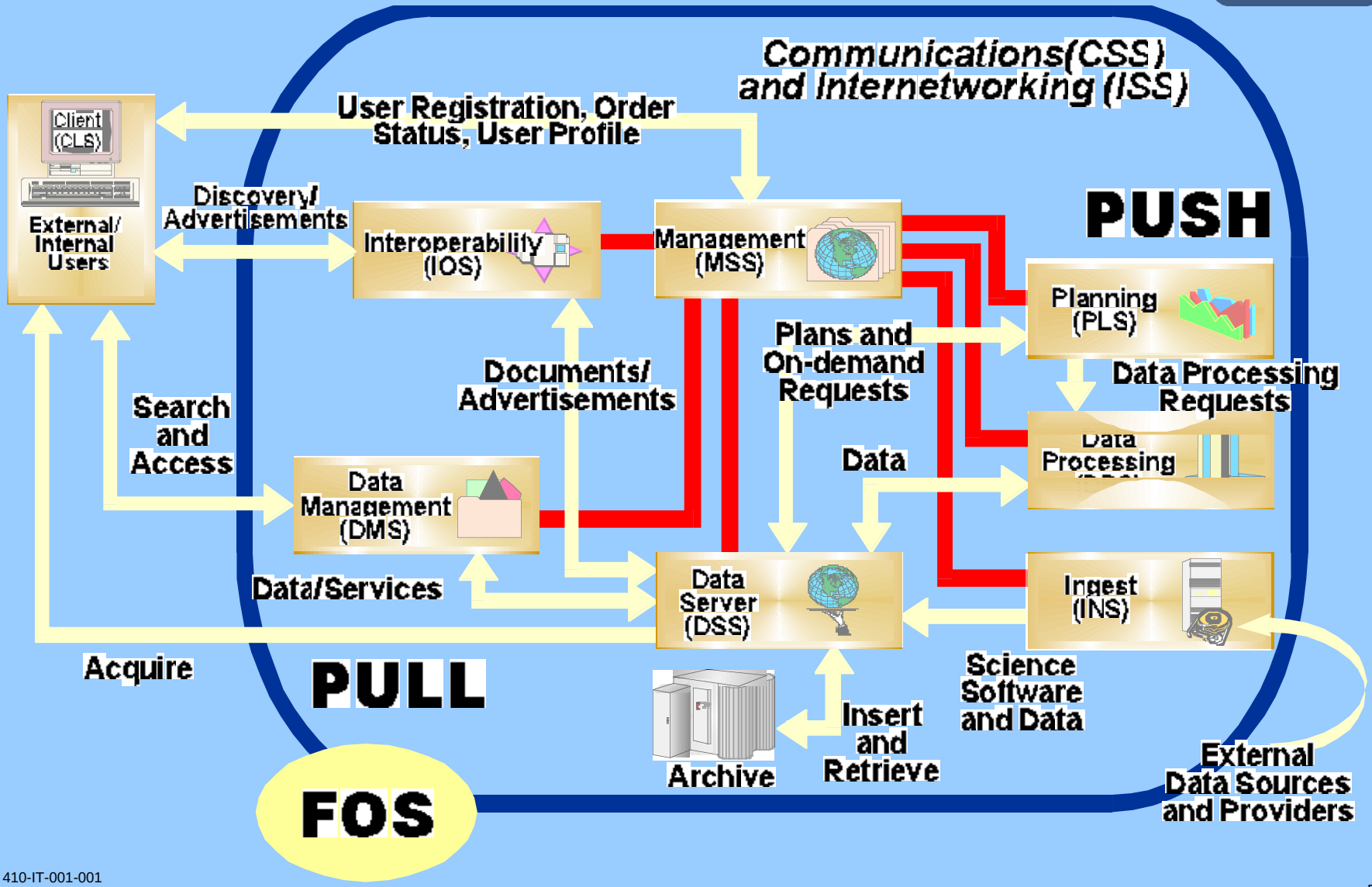


CSS

Doug Dotson

4-5 June 1997

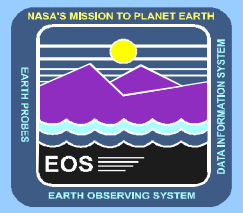
ECS Context


Communications(CSS) and Internetworking (ISS)
User Registration, Order Status, User Profile
Client (CLS)
External/ Internal Users
Discovery/ Advertisements
Interoperability (IOS)
Management (MSS)
PUSH
Planning (PLS)
Plans and On-demand Requests
Data Processing Requests
Documents/ Advertisements
Search and Access
Data Management (DMS)
Data
Data Processing
Data/Services
Data Server (DSS)
Ingest (INS)
Acquire
PULL
Science Software and Data
Insert and Retrieve
Archive
External Data Sources and Providers
FOS

Universal References

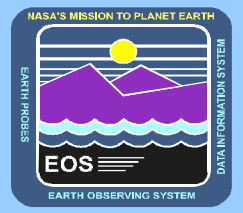
- UR Requirements
- Overview Design

UR Driving Requirements



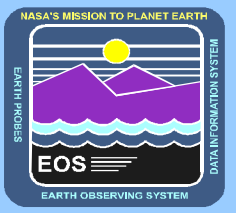
- Provide a mechanism to access ECS object instances through logical references
- Access object instances which can exist anywhere in the system
- Object instances can be any ECS object
- Accommodate nesting of identifiers
  - For example Advertising Service providing references to data/services

UR Driving Requirements



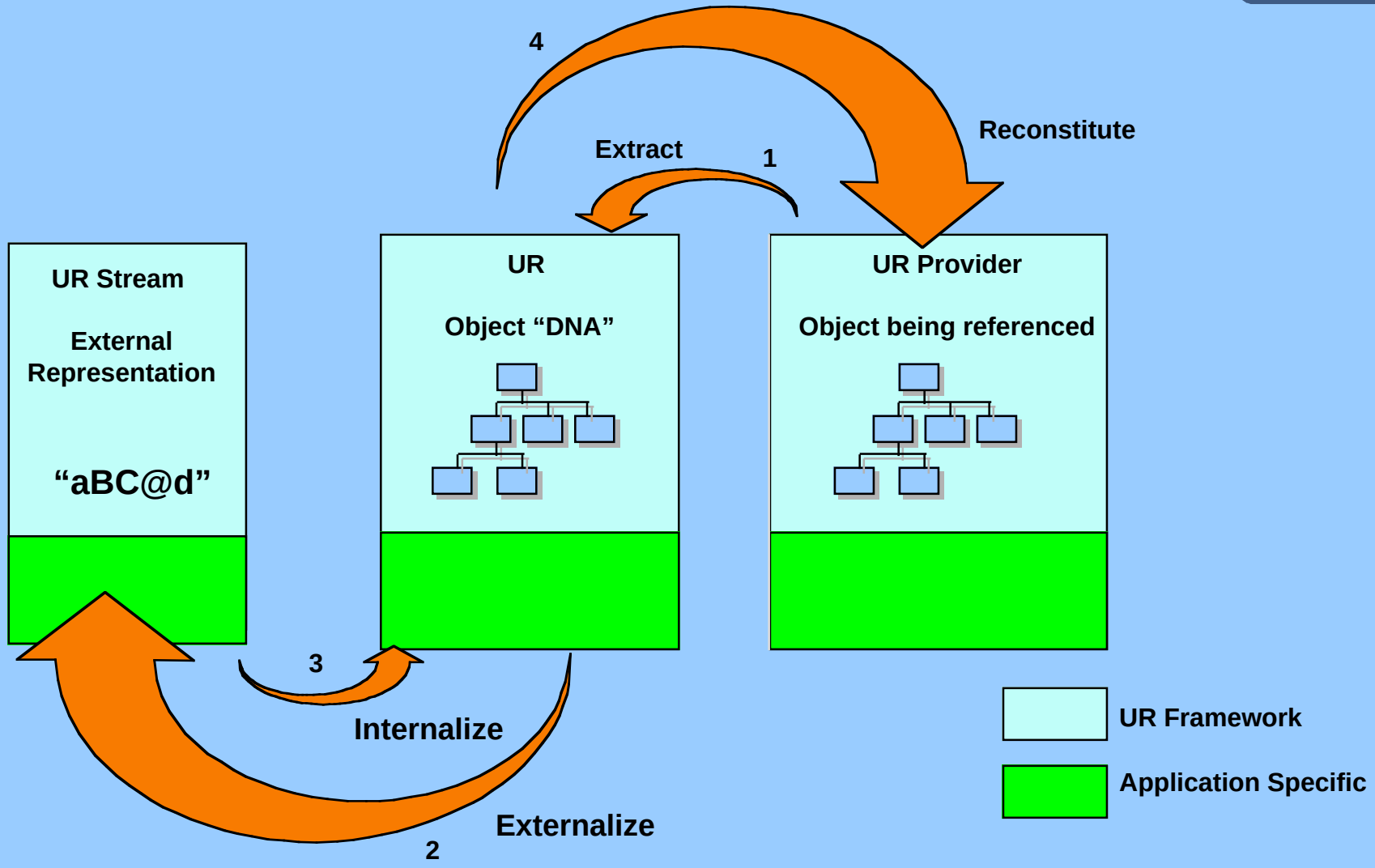
- **Support multiple formats**
  - **C++ objects and ASCII string**
- **Infinite persistence not required**
  - **If object no longer exists, UR is invalid**
- **Unique within application types across DAACs**

UR Solution Overview

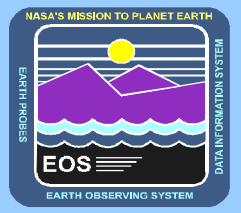


- **Framework Features**
  - **Objects create URs for themselves**
  - **URs used to reconstitute/access original object**
  - **Content of URs is specific to the class of object**
  - **Supports evolution**
    - **Allows abstraction and inheritance to accommodate new objects**
  - **Principles are based on established design patterns (Memento and Object Factory)**

UR Solution Overview



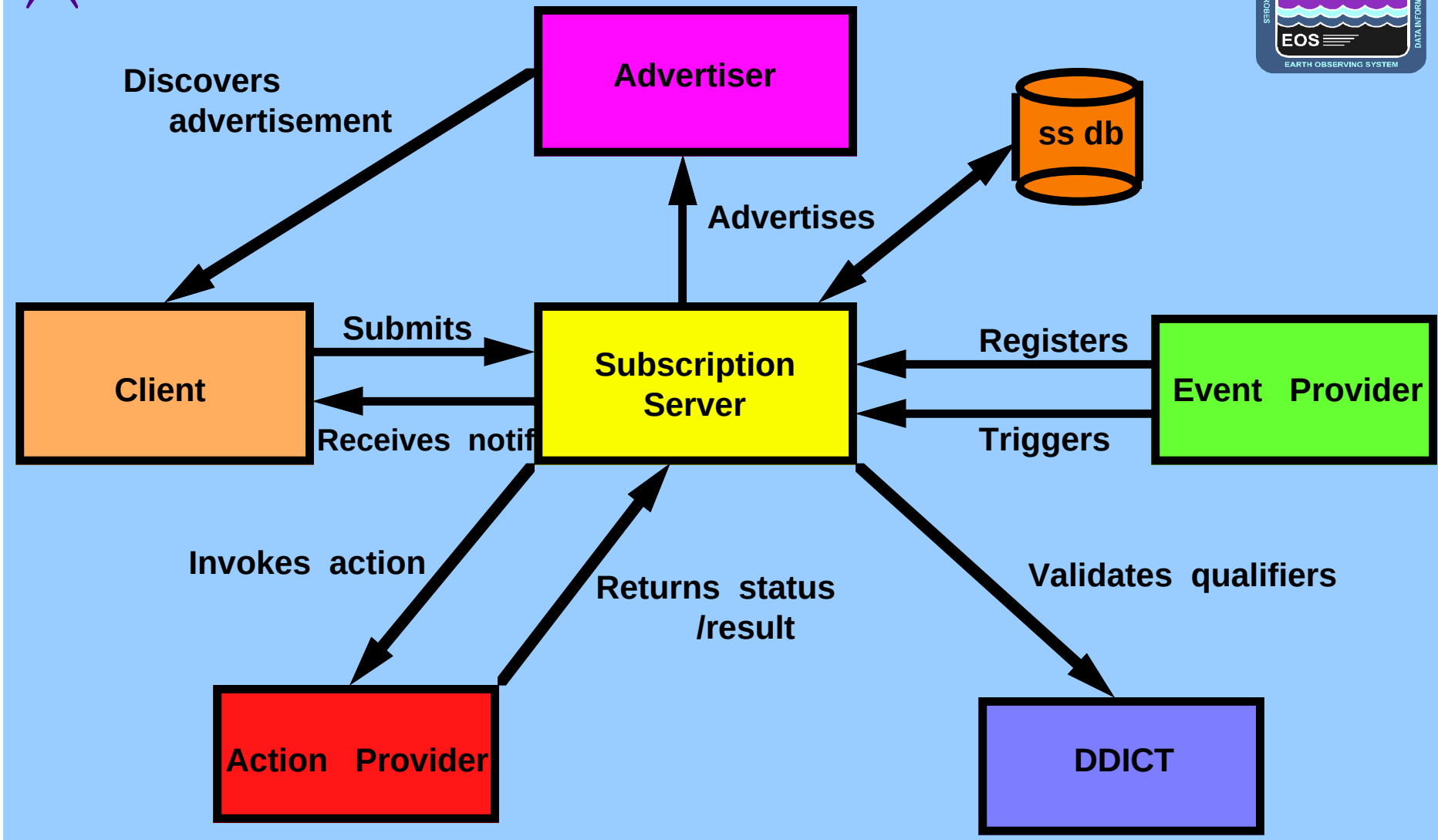
UR Framework Context



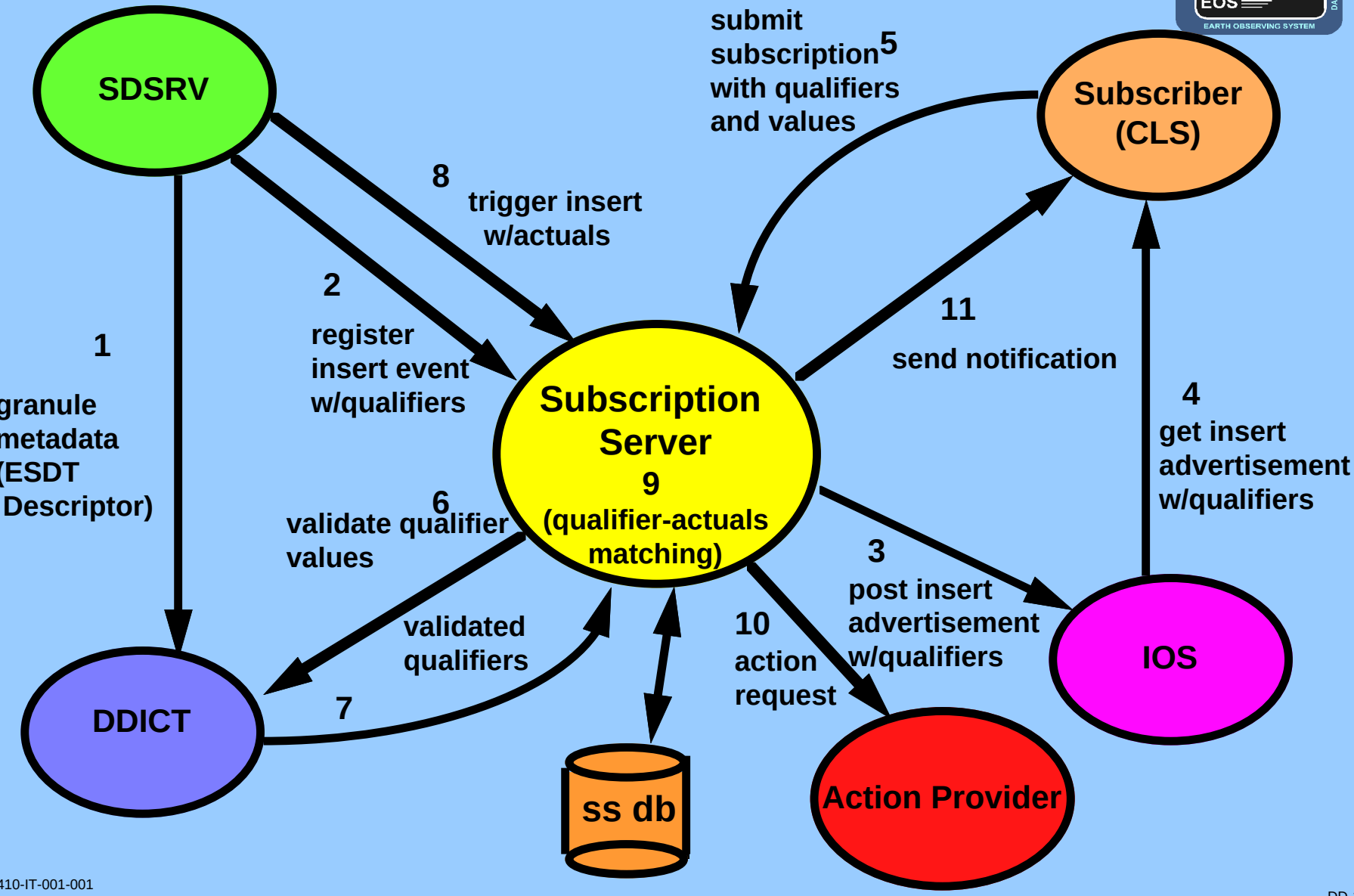
- **Who Uses the UR Framework?**
  - **Earth Science Data/Services e.g., EDSTs**
  - **Clients connecting to servers e.g., Sessions**
  - **Advertisements**
  - **Subscriptions**

Subscription Service

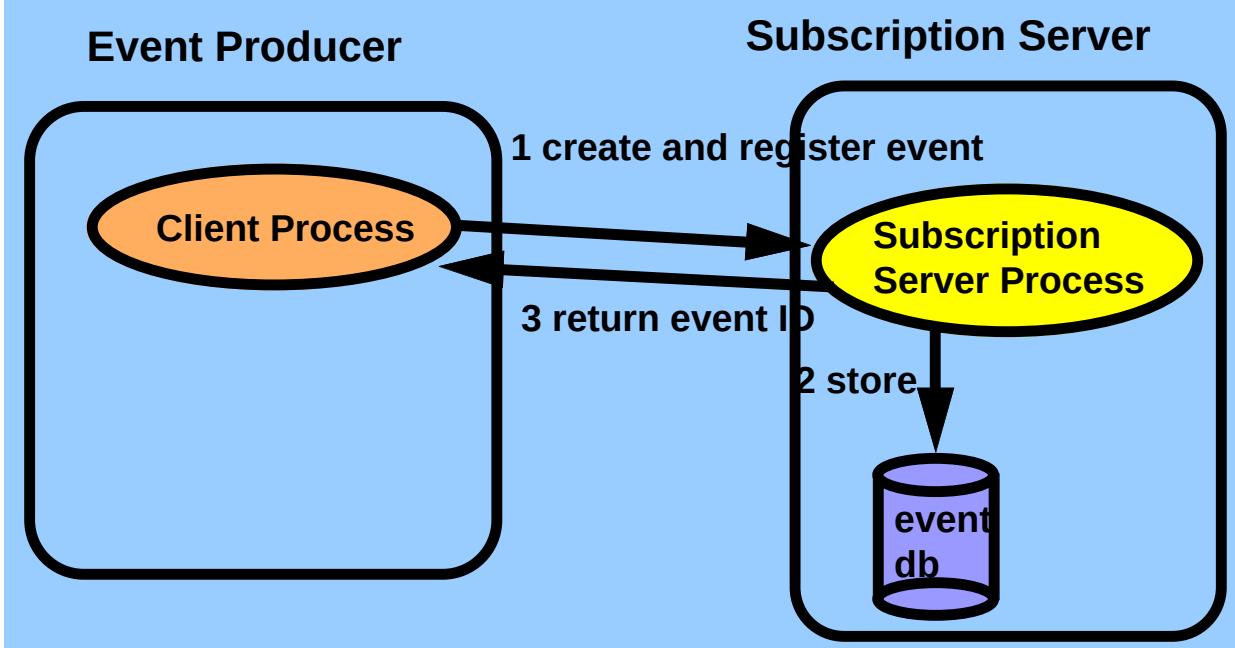
Subscription Server High-level Design



Data Granule Insert Scenario



Register New Event



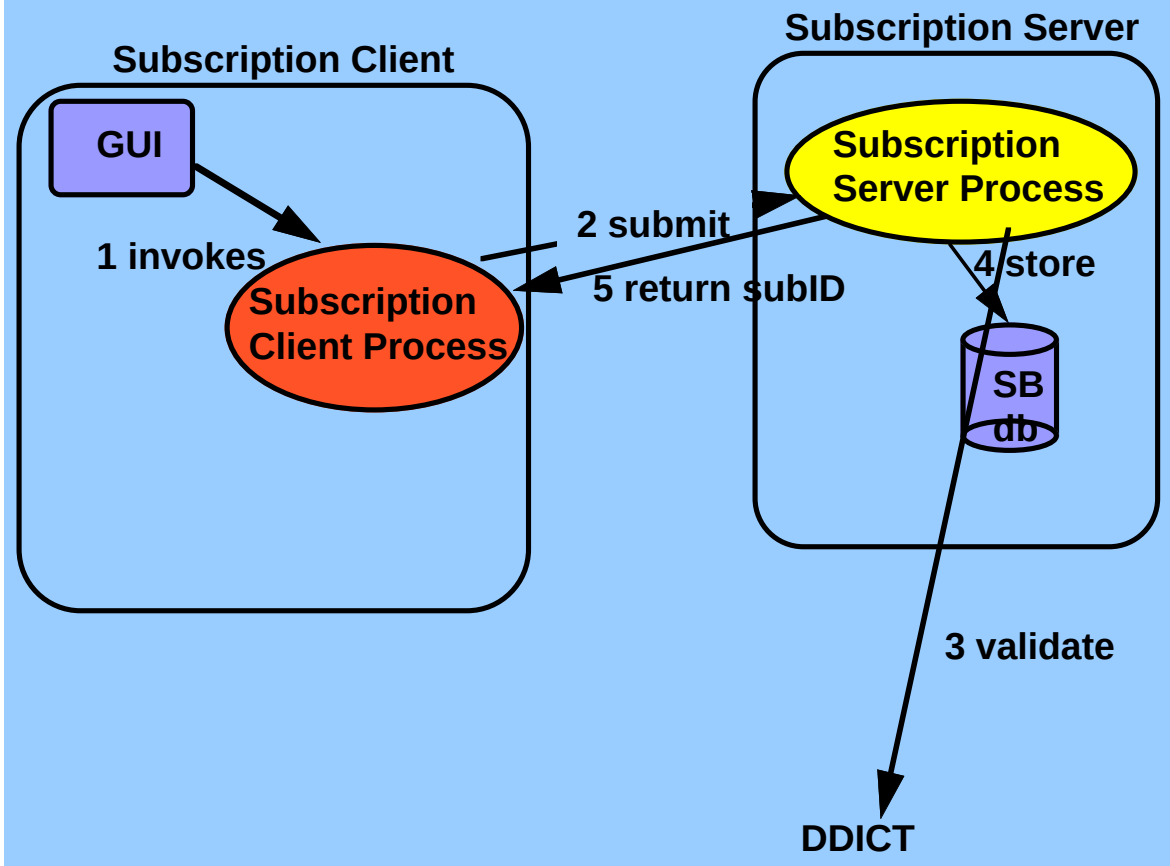
1. Subscription Client process create and register a new event.

2. Subscription server store the event and qualifier names persistantly

3. Subscription server assign a event ID and return to client.

4. Client use GetEventID() to get the ID.

Submitting Subscription



1. A user uses the GUI interface to submit a subscription, providing all necessary data.

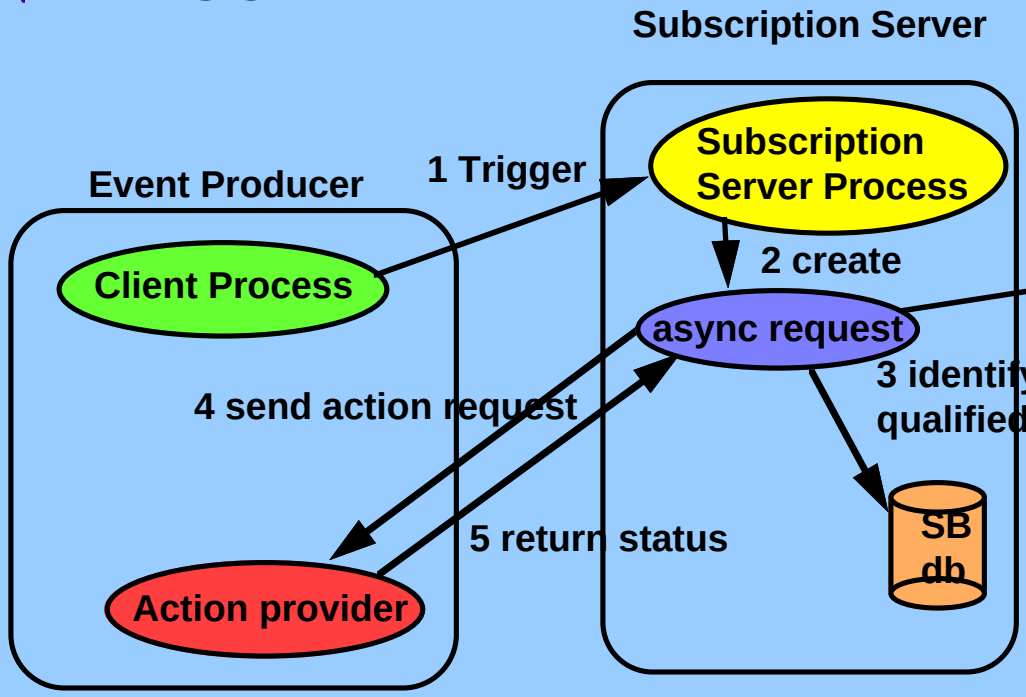
2. The subscription client process is invoked.

3. Subscription server receives the submit request and validates thequalifier.

4. Stores it in a persistant store.

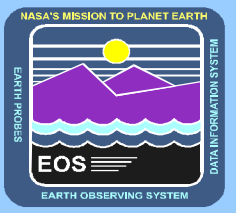
5. Assigns and return a unique subscription ID to subscriber.

Trigger Event



1. Client process instantiates EcClEvent using the event ID. (create a known event) EcClEvent myEvent(EventID); and invokes the trigger method. myEvent.Trigger(GlParameterlist);
2. Subscription Server create an async request object to process subscriptions. Acknownledgement return to client.
3. Subscription Server retrieves qualified subscriptions and matches qualifiers against actuals provided by Event Producer.
4, 5. Process action specified in subscription.
6. Email/IPC notification to subscriber.

Subscription Server Context



- **ECS Subscription Event Producers**
  - **Data Servers**
  - **Advertising Service**
  - **Data Dictionary Service**
- **ECS Subscribers**
  - **Client on behalf of science users**
  - **Planning**
  - **Operator Interfaces on behalf of operators**

Subscription Server Context (continued)

- **Examples of Events:**
  - **Science Granule Insertion**
  - **Metadata Update**
  - **New Advertisement**
  - **New Schema Export to DDICT**
- **Examples of Actions:**
  - **E-mail / IPC**
  - **Acquire data (data push/pull)**